

Отчёт по лабораторной работе №2

Дисциплина: Операционные системы

Мишина Анастасия Алексеевна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выполнение заданий самостоятельной работы	13
4	Выводы	18

Список иллюстраций

2.1	Установка git и gh	6
2.2	Базовая настройка git	7
2.3	Создание ключа ssh по алгоритму rsa и ed25519	7
2.4	Создание ключа gpg	8
2.5	Аккаунт на GitHub	8
2.6	Список ключей	9
2.7	Копирование	9
2.8	Добавление gpg ключа	9
2.9	Настройка автоматических подписей коммитов	10
2.10	Авторизация в gh	10
2.11	Создание новой папки и репозитория	10
2.12	Клонирование репозитория	11
2.13	Новый каталог, удаление лишних файлов, создание каталогов	11
2.14	Команда git add	11
2.15	Команда git commit	11
2.16	Команда git push	12

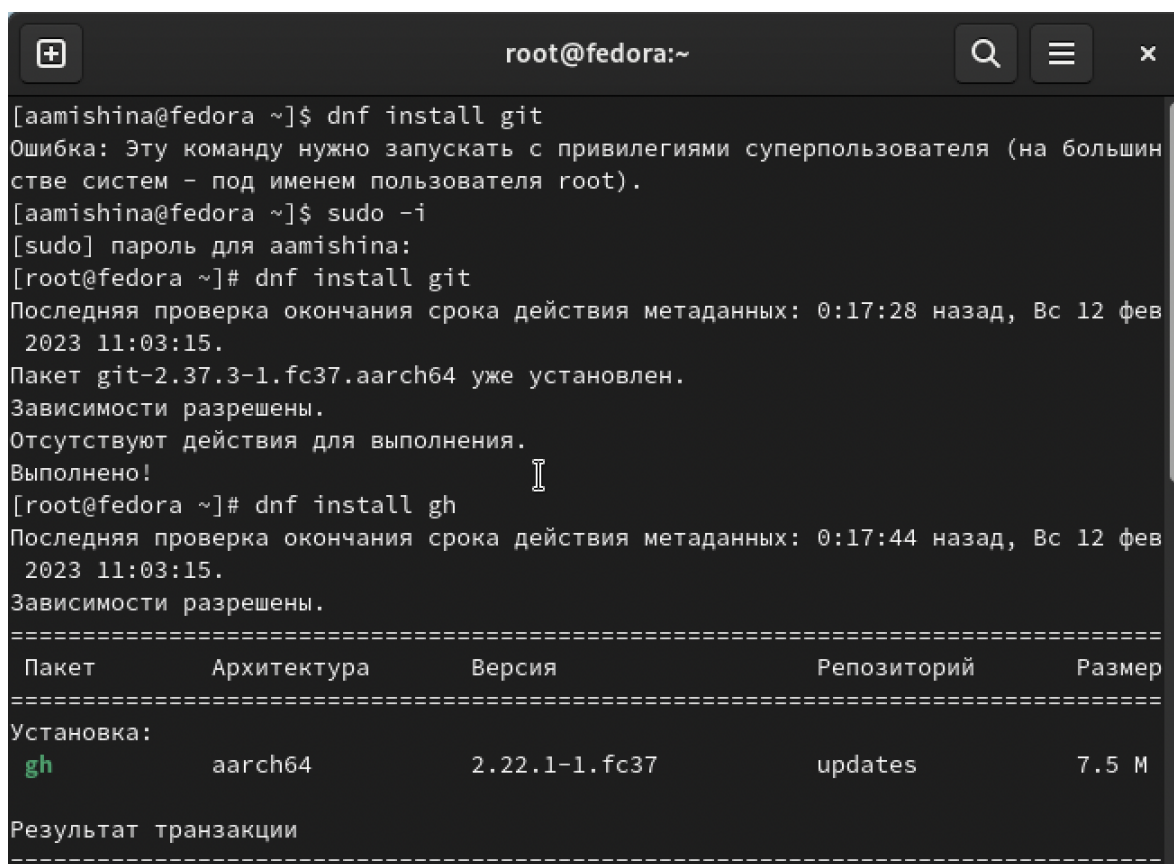
Список таблиц

1 Цель работы

Целью данной работы является изучение идеологии и применение средств контроля версий. А также освоение умений по работе с git.

2 Выполнение лабораторной работы

Для начала требовалось установить git. Делаем это с помощью команды “dnf install git”, также ставим gh (рис. [2.1]).



```
[aamishina@fedora ~]$ dnf install git
Ошибка:Эту команду нужно запускать с привилегиями суперпользователя (на большин
стве систем – под именем пользователя root).
[aamishina@fedora ~]$ sudo -i
[sudo] пароль для аамishина:
[root@fedora ~]# dnf install git
Последняя проверка окончания срока действия метаданных: 0:17:28 назад, Вс 12 фев
 2023 11:03:15.
Пакет git-2.37.3-1.fc37.aarch64 уже установлен.
Зависимости разрешены.
Отсутствуют действия для выполнения.
Выполнено!
[root@fedora ~]# dnf install gh
Последняя проверка окончания срока действия метаданных: 0:17:44 назад, Вс 12 фев
 2023 11:03:15.
Зависимости разрешены.
=====
Пакет      Архитектура  Версия      Репозиторий  Размер
=====
Установка:
gh          aarch64     2.22.1-1.fc37  updates      7.5 М
Результат транзакции
=====
```

Рис. 2.1: Установка git и gh

Затем было необходимо провести базовую настройку git. Задаем имя и email владельца. Настраиваем UTF-8 в выводе сообщений. Задаем имя начальной ветки и работаем с параметрами autocrlf и safecrlf (рис. [2.2])

```
[root@fedora ~]# git config --global user.name "<Анастасия Мишина>"
[root@fedora ~]# git config --global user.email "<nastyam3232@gmail.com>"
[root@fedora ~]# git config --global core.quotepath false
[root@fedora ~]# git config --global init.defaultBranch master
[root@fedora ~]# git config --global core.autocrlf input
[root@fedora ~]# git config --global core.safecrlf warn
[root@fedora ~]#
```

Рис. 2.2: Базовая настройка git

Переходим к созданию ключей. Создаем SSH ключ по алгоритму rsa и размером 4096 бит с помощью команды “ssh-keygen -t rsa -b 4096”. Затем по алгоритму ed25519, пользуясь иной командой “ssh-keygen -t ed25519” (рис. [2.3]).

```
root@fedora:~
[+] 🔍 ☰ ✕
[root@fedora ~]# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:Mzl+0VTHQh8nbumkmmbrwCYz6b8oDN/UV2lYKATVVpc root@fedora
The key's randomart image is:
+---[RSA 4096]-----+
|      .+o. o..=+o|
|      . + .+E=+|
|      o o..=..|
|      ..O+=|
|      .S .oo .|
|      . .+.+.+|
|      + o= =.*|
|      +..* = .|
|      .o.ooo|
+-----[SHA256]-----+
[root@fedora ~]# ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
```

Рис. 2.3: Создание ключа ssh по алгоритму rsa и ed25519

Далее создаем ключи gpg, используя команду “gpg –full-generate-key”. Выбираем опции, указанные в документации на ТУИСе (рис. [2.4]).

```
root@fedora:~  
[root@fedora ~]# gpg --full-generate-key  
gpg (GnuPG) 2.3.7; Copyright (C) 2021 Free Software Foundation, Inc.  
This is free software: you are free to change and redistribute it.  
There is NO WARRANTY, to the extent permitted by law.  
  
gpg: создан каталог '/root/.gnupg'  
gpg: создан щит с ключами '/root/.gnupg/pubring.kbx'  
Выберите тип ключа:  
  (1) RSA and RSA  
  (2) DSA and Elgamal  
  (3) DSA (sign only)  
  (4) RSA (sign only)  
  (9) ECC (sign and encrypt) *default*  
  (10) ECC (только для подписи)  
  (14) Existing key from card  
Ваш выбор? 1  
длина ключей RSA может быть от 1024 до 4096.  
Какой размер ключа Вам необходим? (3072) 4096  
Запрошенный размер ключа - 4096 бит  
Выберите срок действия ключа.  
  0 = не ограничен  
  <n> = срок действия ключа - n дней  
  <n>w = срок действия ключа - n недель  
  <n>m = срок действия ключа - n месяцев
```

Рис. 2.4: Создание ключа gpg

Так как похожу лабораторную работу мы делали в предыдущем семестре, у меня уже есть аккаунт на GitHub, который был настроен ранее (рис. [2.5]).

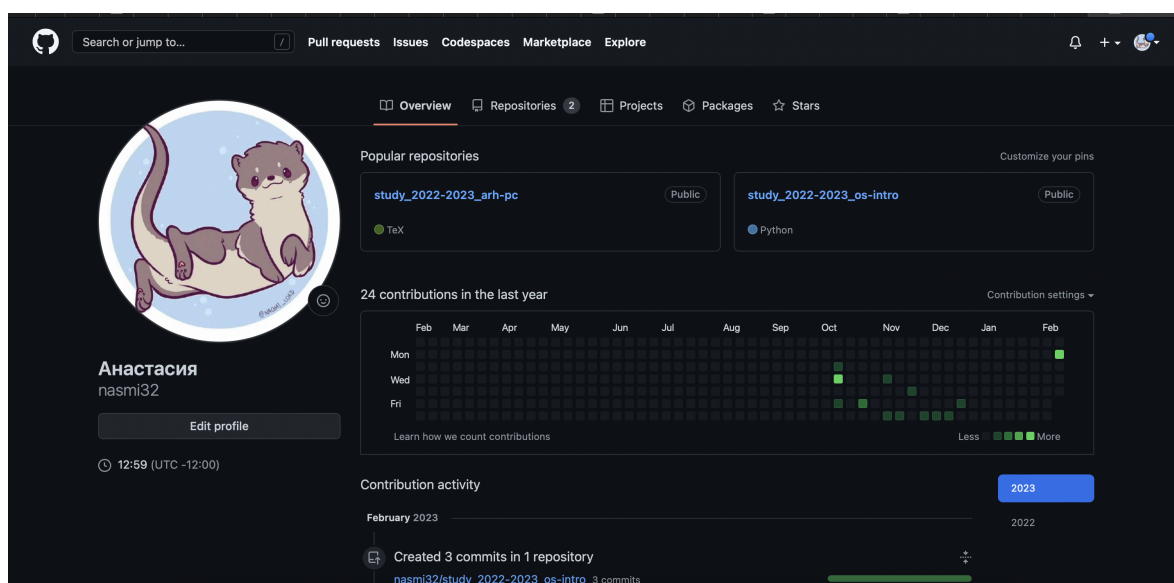


Рис. 2.5: Аккаунт на GitHub

Пришло время добавить ключ gpg на наш аккаунт в GitHub. Выводим список ключей (рис. [2.6]) и копируем отпечаток приватного ключа (рис. [2.7]).

```
[root@fedora ~]# gpg --list-secret-keys --keyid-format LONG
gpg: проверка таблицы доверия
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: глубина: 0 достоверных: 1 подписанных: 0 доверие: 0-, 0q, 0n, 0m, 0f, 1u
/root/.gnupg/pubring.kbx
-----
sec   rsa4096/A3A10D4EE8BD10ED 2023-02-12 [SC]
      382C29ADA1B0218032EF9210A3A10D4EE8BD10ED
uid       [ абсолютно ] Анастасия <nastyam3232@gmail.com>
ssb   rsa4096/1507680DC4FE59F7 2023-02-12 [E]
[root@fedora ~]#
```

Рис. 2.6: Список ключей

```
[root@fedora ~]# gpg --armor --export A3A10D4EE8BD10ED | xclip -sel clip
```

Рис. 2.7: Копирование

Переходим в настройки GitHub, нажимаем на кнопку New GPG key и вставляем полученный ключ в поле ввода (рис. [2.8]).

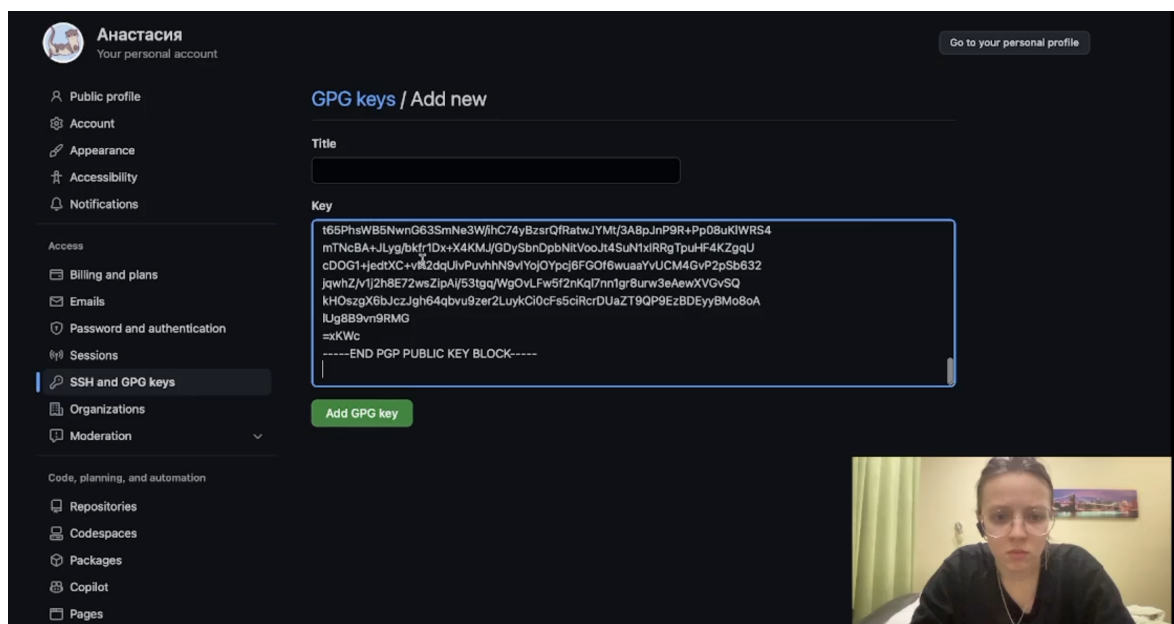


Рис. 2.8: Добавление gpg ключа

Пришло время настроить автоматические подписи коммитов GIT. Используем введенный email и указываем Git применять его при подписи коммитов (рис. [2.9]).

```
[root@fedora ~]# git config --global user.signingkey A3A10D4EE8BD10ED
[root@fedora ~]# git config --global commit.gpgsign true
[root@fedora ~]# git config --global gpg.program $(which gpg2)
[root@fedora ~]#
```

Рис. 2.9: Настройка автоматических подписей коммитов

Авторизуемся в gh (рис. [2.10]).

```
[aamishina@fedora ~]$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Paste an authentication token
Tip: you can generate a Personal Access Token here https://github.com/settings/tokens
The minimum required scopes are 'repo', 'read:org', 'workflow'.
? Paste your authentication token: *****
- gh config set -h github.com git_protocol https
✓ Configured git protocol
✓ Logged in as nasm32
```

Рис. 2.10: Авторизация в gh

Теперь создадим репозиторий курса на основе шаблона. Добавим новую папку “Операционные системы”, создаем репозиторий на основе общего шаблона (рис. [2.11]) и клонируем его (рис. [2.12]).

```
[aamishina@fedora ~]$ mkdir -p ~/work/study/2022-2023/"Операционные системы"
[aamishina@fedora ~]$ cd ~/work/study/2022-2023/"Операционные системы"
[aamishina@fedora Операционные системы]$ gh repo create study_2022-2023_os-intro --template=yamadharma/course-directory-student-template --public
✓ Created repository nasm32/study_2022-2023_os-intro on GitHub
[aamishina@fedora Операционные системы]$
```

Рис. 2.11: Создание новой папки и репозитория

```
[aamishina@fedora Операционные системы]$ git clone --recursive git@github.com:nasmi32/study_2022-2023_os-intro.git
os-intro
Клонирование в «os-intro»...
remote: Enumerating objects: 27, done.
remote: Counting objects: 100% (27/27), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 27 (delta 1), reused 11 (delta 0), pack-reused 0
Получение объектов: 100% (27/27), 16.96 КиБ | 8.48 МиБ/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) зарегистрирован по пути «template/report»
Клонирование в «/home/aamishina/work/study/2022-2023/Операционные системы/os-intro/template/presentation»...
```

Рис. 2.12: Клонирование репозитория

Переходим к настройке каталога курса. Заходим в наш новый каталог “os-intro” и удаляем лишние файлы, создаем необходимые каталоги (рис. [2.13]) и отправляем файлы на сервер (рис. [2.14]), (рис. [2.15], (рис. [2.16])).

```
[aamishina@fedora Операционные системы]$ cd ~/work/study/2022-2023/"Операционные системы"/os-intro
[aamishina@fedora os-intro]$ rm package.json
[aamishina@fedora os-intro]$ echo os-intro > COURSE
[aamishina@fedora os-intro]$ make
```

Рис. 2.13: Новый каталог, удаление лишних файлов, создание каталогов

```
[aamishina@fedora os-intro]$ git add .
```

Рис. 2.14: Команда git add .

```
[aamishina@fedora os-intro]$ git commit -am 'feat(main): make course structure'
```

Рис. 2.15: Команда git commit

```
[aamishina@fedora os-intro]$ git push
Перечисление объектов: 40, готово.
Подсчет объектов: 100% (40/40), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (38/38), 342.42 Киб | 2.50 Миб/с, готово.
Всего 38 (изменений 4), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:nasmi32/study_2022-2023_os-intro.git
   d81f930..7003a2d master -> master
[aamishina@fedora os-intro]$
```

Рис. 2.16: Команда git push

3 Выполнение заданий

самостоятельной работы

Контрольные вопросы:

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначены?

Система контроля версий (от англ. Version Control System, VCS) — программное обеспечение для облегчения работы с изменяющейся информацией.

VCS применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется.

2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.
- Хранилище (репозиторий) — место, где хранятся и поддерживаются какие-либо данные (файлы и история их изменений). Чаще всего

данные в репозитории хранятся в виде файлов, доступных для дальнейшего распространения по сети.

- Commit - фиксация изменений файлов и добавление комментария.
- История - список изменений файлов или данных в проекте (репозитории).
- Рабочая копия - снимок одной версии проекта. Эти файлы извлекаются из сжатой базы данных (хранилища) в каталоге Git и помещаются на диск, для того чтобы их можно было использовать или редактировать.

3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.

Централизованная VCS - клиент-серверная модель: один центральный репозиторий, с которым разработчики взаимодействуют по сети (CVS, Subversion(SVN)).

Децентрализованная VCS - в отличие от централизованной модели может существовать несколько экземпляров репозитория, которые время от времени синхронизируются между собой (GIT, Mercurial).

4. Опишите действия с VCS при единоличной работе с хранилищем.

Разработчик работает с веткой master, также он может создавать новые ветки в проекте. После завершения внесения изменений он коммитит (commit) и пушит (push) их, то есть сохраняет изменения в репозитории.

5. Опишите порядок работы с общим хранилищем VCS.

Каждый разработчик какого-то определенного проекта работает над отдельной частью этого проекта в своей ветке. После завершения внесения изменений, он так же коммитит и пушит их на сервер. После окончания

общей работы необходимо смирджить (merge), т.е. выполнить слияние веток, например, с главной веткой. Также разработчик имеет возможность работать с изменениями другого разработчика, если они работают на одной ветке.

6. Каковы основные задачи, решаемые инструментальным средством git?

Основные задачи: * отслеживание истории изменений * откат изменений * возможность удобной совместной работы над проектом

7. Назовите и дайте краткую характеристику командам git.

Основные команды git:

- Создание основного дерева репозитория: `git init`
- Получение обновлений (изменений) текущего дерева из центрального репозитория: `git pull`
- Отправка всех произведённых изменений локального дерева в центральный репозиторий: `git push`
- Просмотр списка изменённых файлов в текущей директории: `git status`
- Просмотр текущих изменений: `git diff`
- Добавить все изменённые и/или созданные файлы и/или каталоги: `git add .`
- Добавить конкретные изменённые и/или созданные файлы и/или каталоги: `git add имена_файлов`

- Удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): `git rm имена_файлов`
- Сохранить все добавленные изменения и все изменённые файлы: `git commit -am 'Описание коммита'`
- Сохранить добавленные изменения с внесением комментария через встроенный редактор: `git commit`
- Создание новой ветки, базирующейся на текущей: `git checkout -b имя_ветки`
- Переключение на некоторую ветку: `git checkout имя_ветки` (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой)
- Отправка изменений конкретной ветки в центральный репозиторий: `git push origin имя_ветки`
- Слияние ветки с текущим деревом: `git merge --no-ff имя_ветки`
- Удаление локальной уже слитой с основным деревом ветки: `git branch -d имя_ветки`
- Принудительное удаление локальной ветки: `git branch -D имя_ветки`
- Удаление ветки с центрального репозитория: `git push origin :имя_ветки`

8. Что такое и зачем могут быть нужны ветви (branches)?

Под веткой принято понимать независимую последовательность коммитов в хронологическом порядке. Однако конкретно в Git реализация

ветки выполнена как указатель на последний коммит в рассматриваемой ветке. После создания ветки уже новый указатель ссылается на текущий коммит

Ветки используются для разработки одной части проекта отдельно от других. Они позволяют работать одновременно над разными версиями проекта.

9. Как и зачем можно игнорировать некоторые файлы при commit?

Во время работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять в последствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл `.gitignore` с помощью сервисов.

4 Выводы

В ходе выполнения данной лабораторной работы я приобрела практические навыки по работе с git. Изучила идеологию и применение средств контроля версий. Также мне удалось составить отчет, прикрепив скриншоты, которые я делала во время выполнения задания.